배열

1 배열(Array)

1 같은 타입의 요소들이 연속해서 저장 되어있는 데이터 모임

➡ **타입 T**와 **상수 N**이 주어졌을 때,
타입 T[N]의 변수는 타입 T인 N개의 요소들의 배열을 갖는다.
배열의 각 요소는 0 부터 N-1 까지의 **색인(index)**에 의해 참조된다.

2 예시(index 통한 참조)

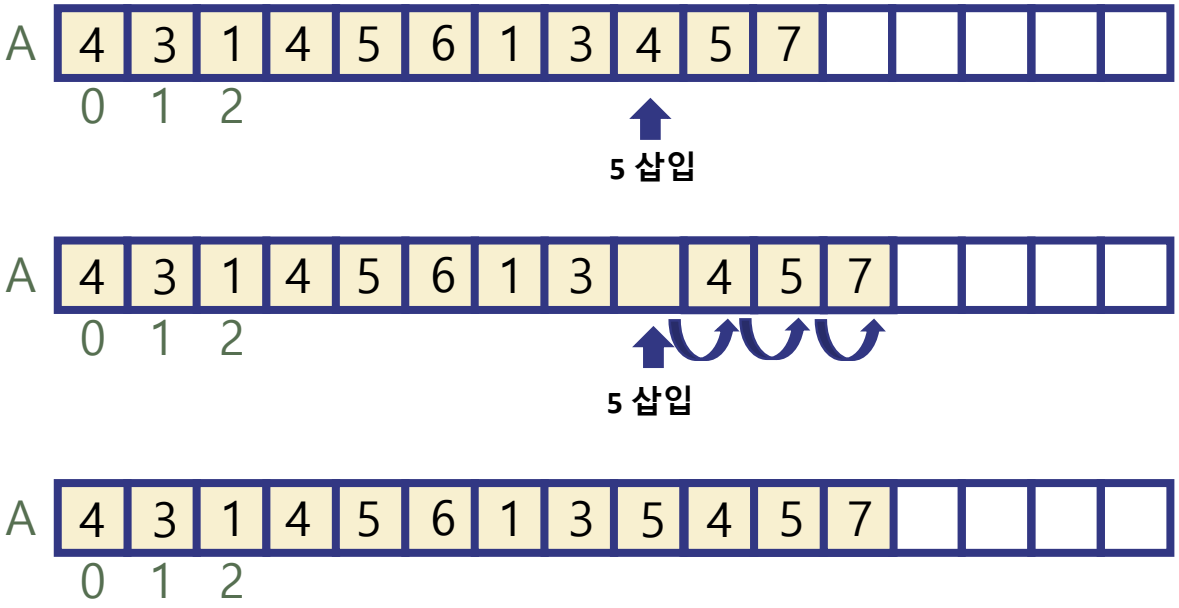
```
double f[5];        // 5개의 double 배열 ; f[0],...,f[4]
int m[10] ;         // 10개의 int 배열 : m[0],...,m[9]

m[2]=4;
f[4] = 2.5;
cout << f[m[2]];  // f[4]의 값 2.5를 출력
```

3 단점

➡ 미리 **크기**를 정해야 하고, 한 번 정한 **크기는 변경할 수 없다**.
➡ **삽입**과 **제거**에 있어 상대적으로 cost가 크다

5 배열의 삽입

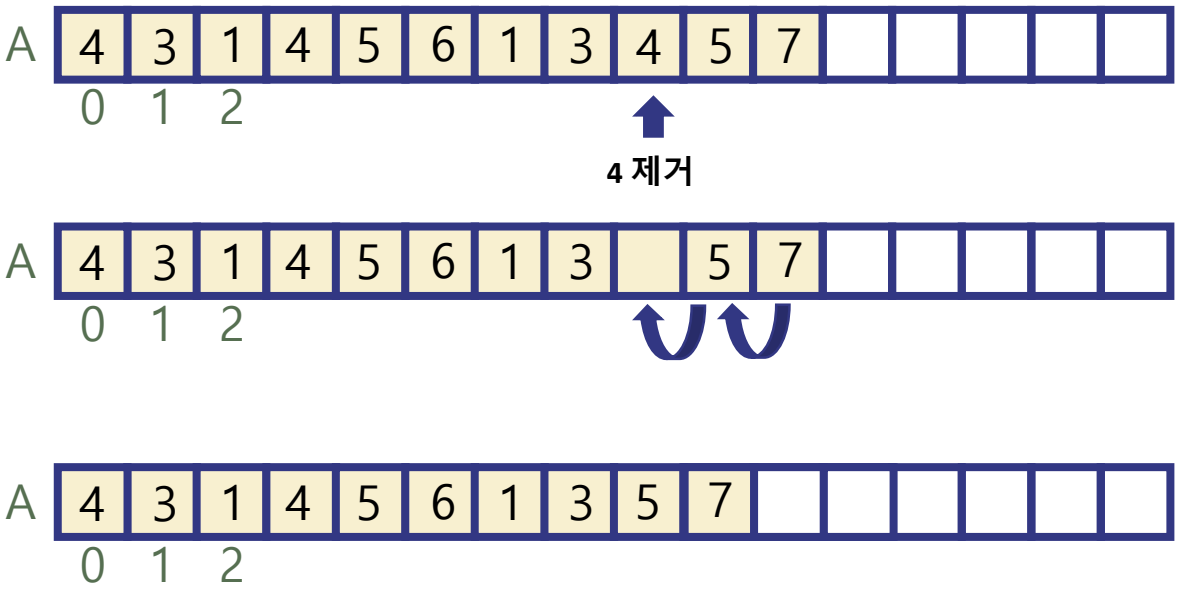
- i 번째 위치에 원하는 데이터 넣고 싶다 **add(i,element)**
- 순차,연속적으로 저장 되어 있는 형태를 유지 하고 싶다



- 최악의 경우 O(n)

4 배열의 제거

- i 번째 위치에 데이터 제거하고 싶다 **remove(i)**
- 순차,연속적으로 저장 되어 있는 형태 유지 하고 싶다



- 최악의 경우 O(n)

배열을 이용한 기본 연산

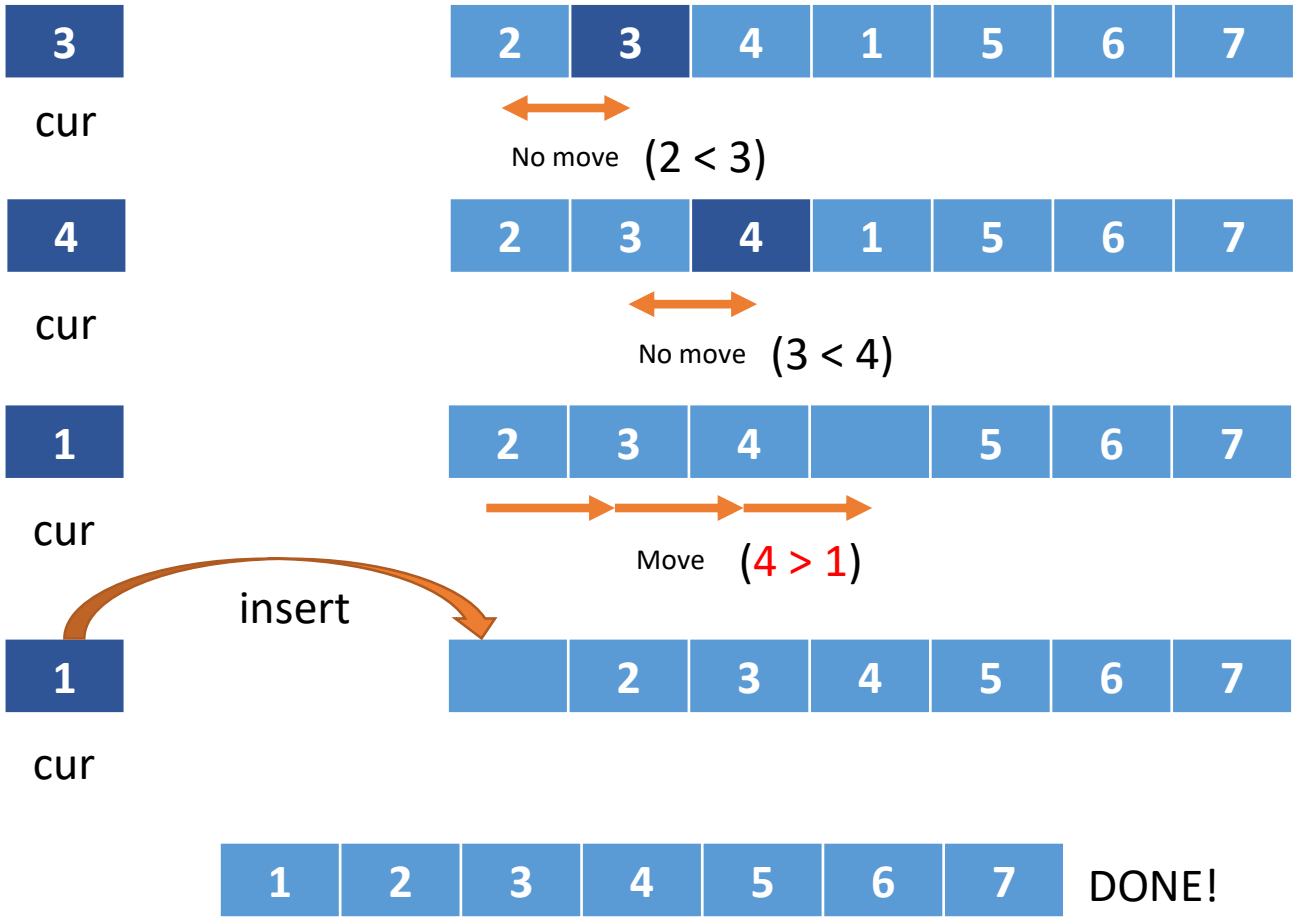
- at : 인덱스 i 를 통해서 데이터에 접근
- set : 인덱스 i 저장된 값을 원하는 값으로 덮어씌움
- add : 인덱스 i 에 원하는 값을 삽입
- remove : 인덱스 i 값을 삭제
- printArray : 배열 내부의 원소들을 순차적으로 출력

```
class Array {
public:
    int n;      // 배열 내 원소 수
    int* arr;   // 배열

    Array(int size) {
        this->n = 0;                        // 배열 내 원소 수 0으로 초기화
        this->arr = new int[size] ;         // 크기가 size인 배열 할당
        for (int i = 0; i < size; i++) {    // 배열 원소 0으로 초기화
            arr[i] = 0;
        }
    }

    int at(int idx) {  // 해당 index에 저장된 값 반환
                                // 원소가 없는 index값 주어졌을 때


    }
```

```
void set(int idx, int X) { // 해당 index에 저장된 값 덮어씌우기
    if           {

    }
    else {                 // 원소가 없는 index값 주어졌을 때

    }
}

void add(int idx, int num) { // 해당 index에 값 삽입
    if            {     // 원소가 없는 index값 주어졌을 때


    }
    else {


    }
}
```

```
int remove(int idx) {  // 해당 index에 값 삭제하고 그 값 반환
    if                          // 원소가 없는 index값 주어졌을 때
    int value = arr[idx];









    return value;
}

void printArray() {    // 배열에 저장되어 있는 값 차례로 출력
    if                                  // 배열이 비었을 때
    else {



    }
}
```

5 배열을 이용한 정렬(오름차순)




Thank you